

**PROGRAMMATION AVANCÉE ET ALGORITHMIQUE**

*-Travaux Pratique 4-*

***Réalisé par :***

Chao PAN

***Encadrée par :***

M. Renaud Marlet et M. Pascal Monasse

***Octobre 2019***

TP4

2.1

Le parcours en profondeur d’abord (pre-order) correspond la suite 12 8 4 9 23 17 15

3.1

Tous les cas d’erreur pour les fonctions de IntTree :

1) IntTree\* getSon(int pos) : pos >= sons.size()

2) void setSon(int pos, IntTree\* newSon : pos >= sons.size()

3) void removeLastSon() : sons.size() == 0

3.2

Pour la function de getSon(), on peut retourner un pointeur vide lors d’une erreur. Pour la fonction setSon et celle-ci removeLastSon, on peut facilement ajouter un statut d'erreur

3.3

Pour les 3 fonctions, on peut signaler l’erreur par exception.

3.4

Un mode de signalement d’erreur : rattrapage d’exception.

4.2

Non. Parce que le mécanisme de template se concentre à créer une fonction pour chaque type de variable, par conséquent, il est indispensable de connaitre le type de variables ainsi que la définition de fonction. C’est-à-dire qu’il faut mettre la déclaration et la définition en même fichier (.h).

4.3

Oui. Parce qu’on a rajouté deux nouvelles fonctions (insertSon et removeSon), pour eux, il faut compléter la gestion d’erreur. En revanche, pour d’autre fonctions, ce n’est pas changer la gestion d’erreur.

5.2

Pour calculer la profondeur maximale d'un arbre, c’est mieux de prendre une fonction récursive en profondeur d'abord, parce que cette fonction n’a pas besoin d’allocation de mémoire, c’est-ç-dire, on n’a pas besoin d’allouer de mémoire pour chaque nœud d’arbre.

5.3

Pour calculer la profondeur minimale d’un arbre, c’est mieux de prendre une fonction non-récursive en largeur d’abord, parce que cette fonction marche plus vite. En effet, dans un arbre quand on arrive à un nœud qui n’a aucun enfant, la distance entre ce nœud et celui de racine est la minimal, donc, c’est mieux de parcourir couche par couche.